FIG. 1

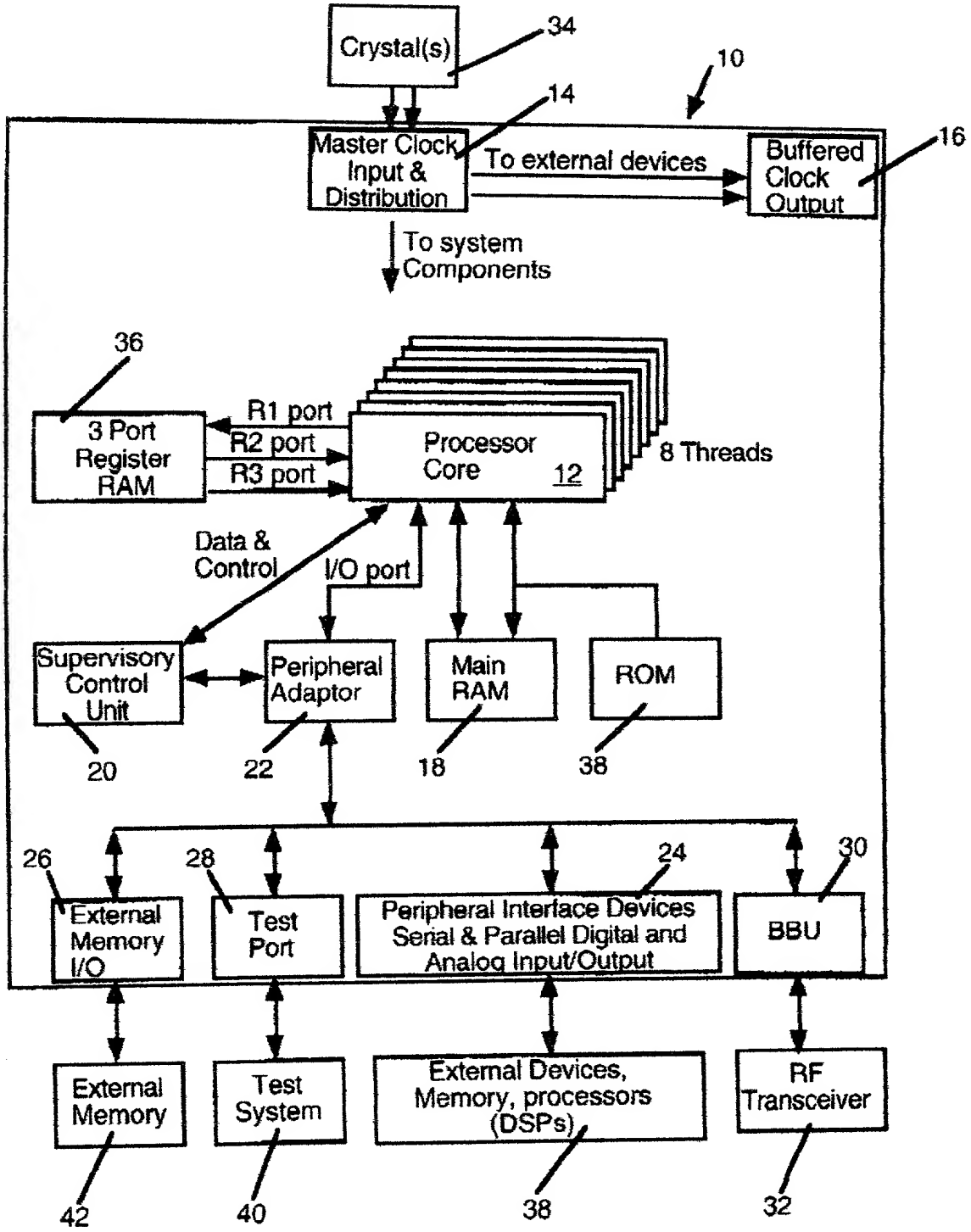
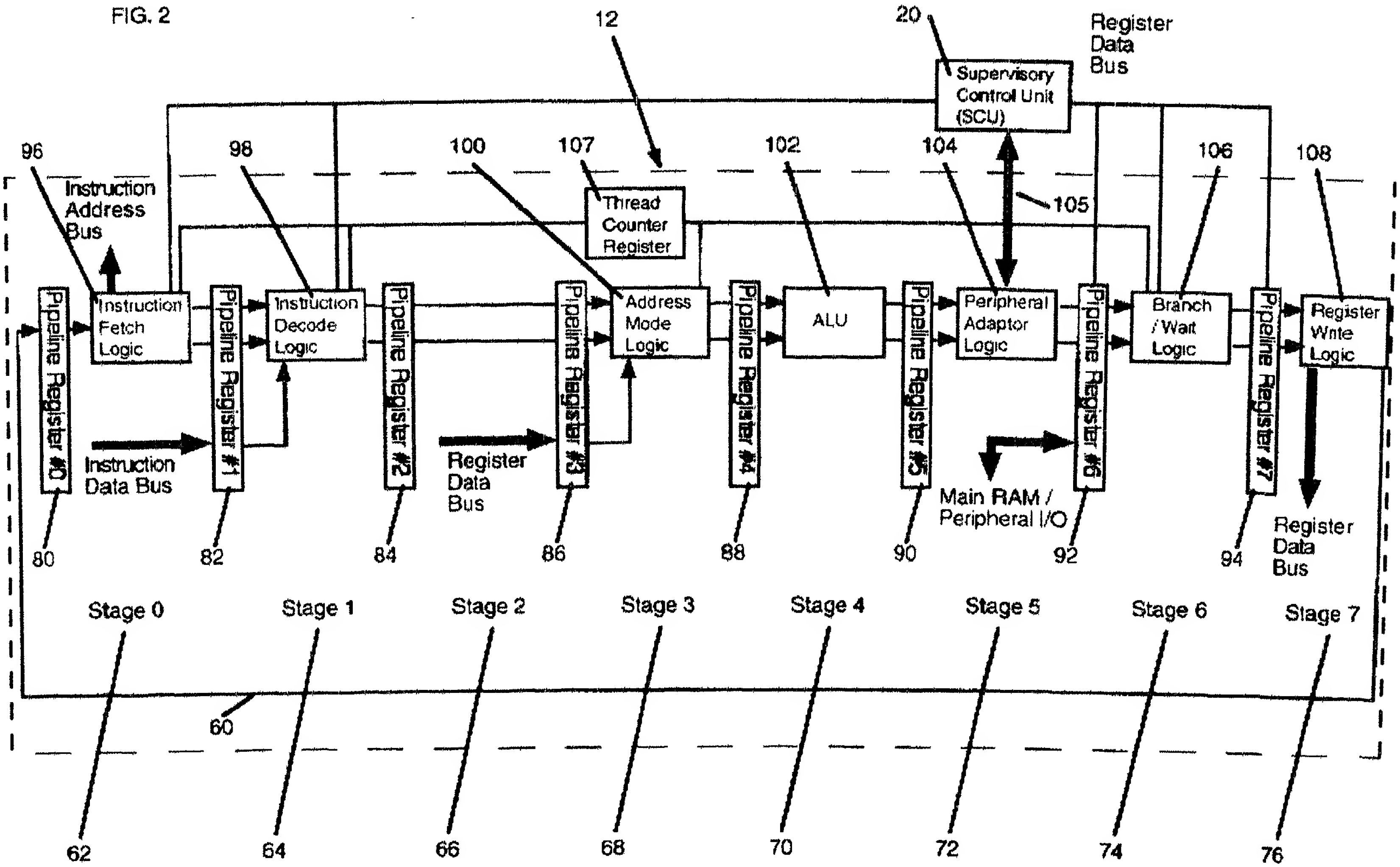

Crystal(s)
34
10
14
Master Clock Input & Distribution
To external devices
Buffered Clock Output
16
To system Components
36
3 Port Register RAM
R1 port
R2 port
R3 port
Processor Core 12
8 Threads
Data & Control
I/O port
Supervisory Control Unit
Peripheral Adaptor
Main RAM
ROM
20
22
18
38
26
28
24
30
External Memory I/O
Test Port
Peripheral Interface Devices Serial & Parallel Digital and Analog Input/Output
BBU
External Memory
Test System
External Devices, Memory, processors (DSPs)
RF Transceiver
42
40
38
32

FIG. 2



12
20
Supervisory Control Unit (SCU)
Register Data Bus
96
Instruction Address Bus
98
100
107
Thread Counter Register
102
104
105
106
108
Pipeline Register #0
Instruction Fetch Logic
Pipeline Register #1
Instruction Decode Logic
Pipeline Register #2
Pipeline Register #3
Address Mode Logic
Pipeline Register #4
ALU
Pipeline Register #5
Peripheral Adaptor Logic
Pipeline Register #6
Branch / Wait Logic
Pipeline Register #7
Register Write Logic
Instruction Data Bus
Register Data Bus
Main RAM / Peripheral I/O
Register Data Bus
80
82
84
86
88
90
92
94
Stage 0
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6
Stage 7
60
62
64
66
68
70
72
74
76

FIG. 3

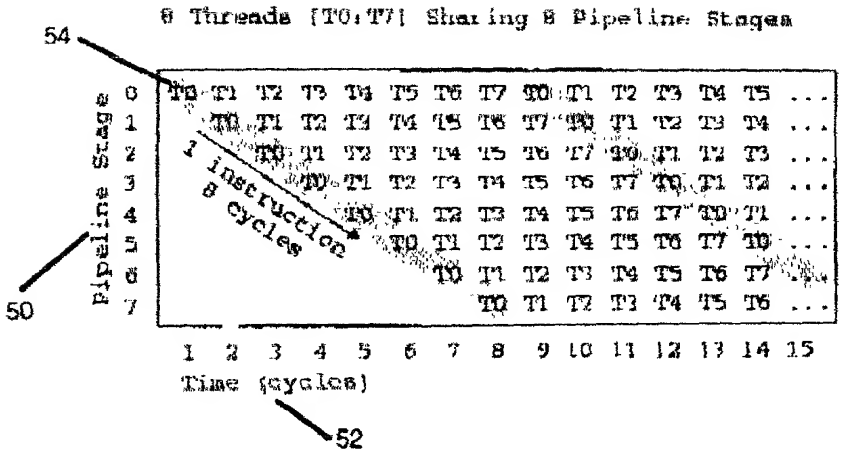


FIG. 4

PIPELINE STAGE		RESOURCE USAGE - Processor Logic or System Memory								
		Instruction Fetch Logic	ROM or 2 Port Main RAM	Instruction Decode Logic	Register RAM (3port)	Address Mode Logic	ALU	Peripheral Adaptor Logic	Branch / Wait Logic	Register Write Logic
Stage #	Description									
0	Instruction Fetch	Used	Read							
1	Instruction Decode			Used						
2	Register Reads				Read	Used				
3	Address Modes						Used			
4	ALU Operation									
5	Memory or I/O Cycle		Read or Write					Read or Write	Used	
6	Branch/Wait									Used
7	Register Write				Write					

58 56

FIG. 5

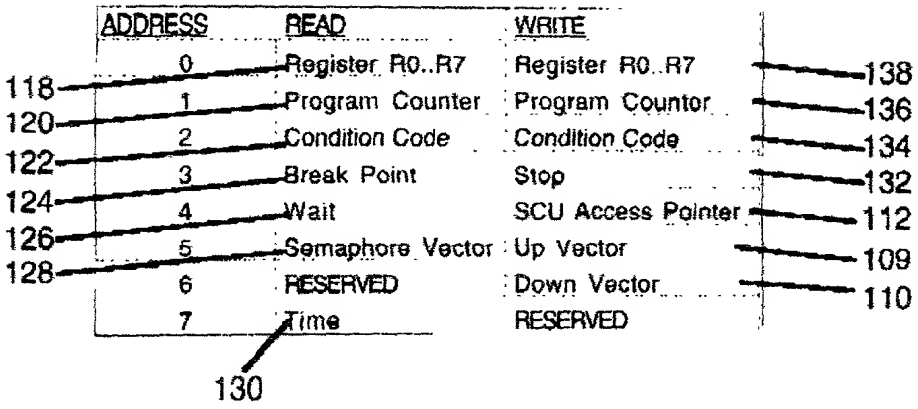


FIG. 6

FIG. 7

140 Address Mode	142 Description	143 1-Word	2-Word
register	Rn	yes	no
register indirect	*Rn	yes	no
base displacement	*(Rn+K)	yes	yes
PC relative	*(PC+K)	yes	yes
absolute	*K	no	yes
immediate	K	some	some

FIG. 8

Instr	Description	Available Address Modes
add	2's complement add	register, immediate
and	bitwise and	register, immediate
bc	conditional branch	PC relative
bic	bit clear	immediate
bis	bit set	immediate
bix	bit change	immediate
bra	unconditional branch	PC relative
inp	read input port	immediate
ior	bitwise inclusive or	register, immediate
jsr	jump to subroutine	register indirect, absolute
ld	load from RAM	base displacement, absolute
mov	move immediate	immediate
outp	write output port	immediate
rol	bitwise rotate left	register, immediate
st	store to RAM	base displacement, absolute
sub	2's complement subtract	register
thrd	get thread number	register
xor	bitwise exclusive or	register, immediate

146

FIG. 9

```
                    // Initialize Constants
150                 SCUptr 0x04          // SCU pointer register
                    SCUpc  0x00          // SCU program counter register
                    SCUreg 0x00          // SCU thread register register
                    SCUsr  0x03          // SCU stop/run register
                Word
151             Address
                0 // System powers up in SIMD mode with all threads using common code
                0
152             1 InitializeThreads:     // ALL THREADS RUNNING
                1
                1 thrd   r0              // differentiate threads
                2 mov    r2, 0x00        // initialize register R2 to zero
                3
154             3 InitMemory:            // write zeros to memory, SIMD Mode
                3
                3 st     r2, r0, 0x00    // 8-way parallel store to memory
                4 add    r0, r0, 0x08    // move threads to next 8 memory locations
                5 blc    r0, r0, 0x0E    // Check if 16k words initialized by testing bit 14 of word
                6 bc     0x9, Initmemory // if v bit=0, branch back
                7
156             7 StopThreads:           // stop threads 1 to 7
                7
                7 mov    r7, 0xFE        // set up mask to only select thread 0
                8 outp   r7, SCUsr       // set SCU stop vector for only thread 0 running
                9
158             9 InitForMIMD:           // ONLY THREAD ZERO RUNNING
                9
                9 mov    r5, 0x38        // select SCU pointer value for thread 7 & its register R0
               10 mov    r6, 23          // set pointer to start of MIMD branch table
               12 mov    r7, 0x800       // branch location for thread 7
               14 mov    r0, 0x100       // point thread to its branch location
               15
160            15 SetMIMD:              // restart threads in MIMD operating mode
               15
               15 outp   r5, SCUptr      // select thread to change SCU pointer register
               16 outp   r6, SCUpc       // initialize program counters by SCU PC register
               17 outp   r7, SCUreg      // initialize R0 of selected thread to MIMD branch location
               18 sub    r7, r7, 0x100   // pointer to next branch address
               19 sub    r5, r5, 0x08    // shift to next thread value
               20 bc     0x0A, SetMIMD   // loop until program counters of thread 7 to 1 initialized
               21 mov    r4, 0x00        // set up SCU mask to select all threads
               22 outp   r4, SCUsr       // set SCU stop vector to run all threads
               23
162            23 MIMDStart:            // each thread branches to individual independent programs
               23
               23 jsr    r0, r0          // jump to different program for each thread, start MIMD
```